BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G09G 5/02

A1

(11) International Publication Number: WO 96/01466

(43) International Publication Date: 18 January 1996 (18.01.96)

(21) International Application Number: PCT/US95/08257

(22) International Filing Date: 3 July 1995 (03.07.95)

(30) Priority Data:
08/269,982 1 July 1994 (01.07.94) US

(71) Applicant: APPLE COMPUTER, INC. [US/US]; One Infinite Loop - MS 38-PAT, Cupertino, CA 95014 (US).

(72) Inventors: SWEN, Iue-Na; 22395 St. Andrews Avenue, Cupertino, CA 95014 (US). STOKES, Michael, D.; 22462 Salem Avenue #3, Cupertino, CA 95014 (US). MOHR, Thomas, E.; 555 N.W. Park #407, Portland, OR 97209 (US).

(74) Agents: SIMON, Nancy, R. et al.; Apple Computer, Inc., 1 Infinite Loop - MS: 38-PAT, Cupertino, CA 95014 (US).

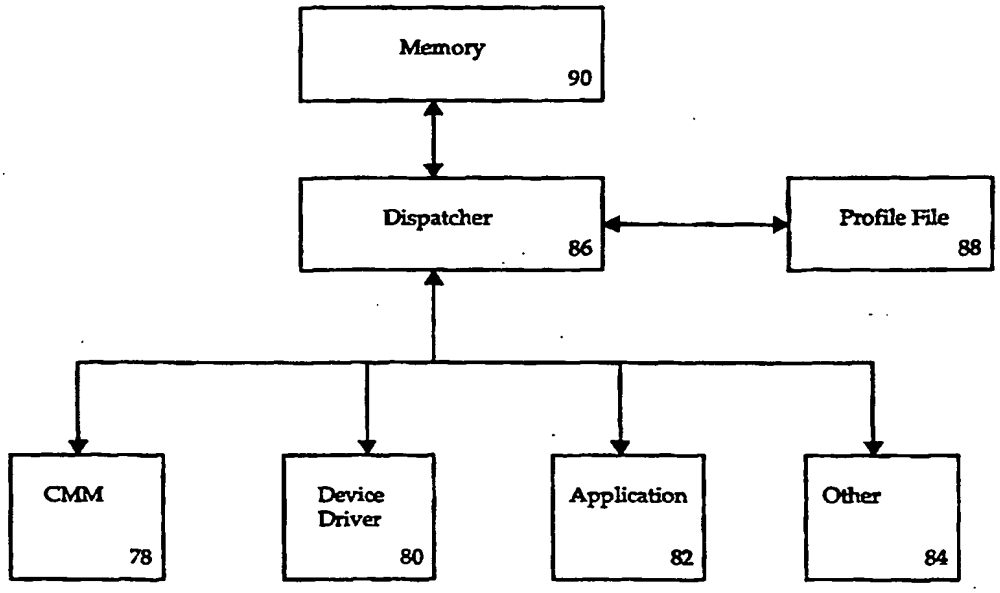
(81) Designated States: AM, AT, AU, BB, BG, BR, BY, CA, CH, CN, CZ, DE, DK, EE, ES, FI, GB, GE, HU, IS, JP, KE, KG, KP, KR, KZ, LK, LR, LT, LU, LV, MD, MG, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, TJ, TM, TT, UA, UG, UZ, VN, European patent (AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG), ARIPO patent (KE, MW, SD, SZ, UG).

Published

With international search report.

Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: METHOD AND SYSTEM FOR DYNAMICALLY DISPATCHING COLOR PROFILE DATA IN A COLOR MANAGEMENT SYSTEM



(57) Abstract

A modular architecture of a color management system provides for multiple accesses of a device profile and manipulation of the device profile, or portions of the device profile, while maintaining the integrity of the device profile. The device profile structure is defined as a series of tagged elements that can be accessed randomly and individually.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AT	Austria
AU	Australia
BB	Barbados
BE	Belgium
BF	Burkina Faso
BG	Bulgaria
BJ	Benin
BR	Brazil
BY	Belarus
CA	Canada
CF	Central African Republic
CG	Congo
CH	Switzerland
CI	Côte d'Ivoire
CM	Cameroon
CN	China
CS	Czechoslovakia
CZ	Czech Republic
DE	Germany
DK	Denmark
ES	Spain
FI	Finland
FR	France
GA	Gabon
GB	United Kingdom
GE	Georgia
GN	Guinea
GR	Greece
HU	Hungary
IE	Ireland
IT	Italy
JP	Japan
KE	Kenya
KG	Kyrgystan
KP	Democratic People's Republic of Korea
KR	Republic of Korea
KZ	Kazakhstan
LI	Liechtenstein
LK	Sri Lanka
LU	Luxembourg
LV	Latvia
MC	Monaco
MD	Republic of Moldova
MG	Madagascar
ML	Mali
MN	Mongolia
MR	Mauritania
MW	Malawi
NE	Niger
NL	Netherlands
NO	Norway
NZ	New Zealand
PL	Poland
PT	Portugal
RO	Romania
RU	Russian Federation
SD	Sudan
SE	Sweden
SI	Slovenia
SK	Slovakia
SN	Senegal
TD	Chad
TG	Togo
TJ	Tajikistan
TT	Trinidad and Tobago
UA	Ukraine
US	United States of America
UZ	Uzbekistan
VN	Viet Nam

METHOD AND SYSTEM FOR DYNAMICALLY DISPATCHING COLOR PROFILE DATA IN A COLOR MANAGEMENT SYSTEM

The present invention relates in general to computer systems, and in particular to color management systems operating in computer systems. Still more particularly, the present invention relates to a method and system for dynamically dispatching color profile data in a color management system.

Color management refers to a system that supports color information exchange, color matching, profile management, and device calibration. As color input, display and output devices for computer systems proliferate, the need for color management increases. This is due to several factors. First, differing device types operate in different color spaces. For example, color display monitors typically display colors as combinations of red, green, and blue, and are said to work in the RGB color space. Printers typically work print images as combinations of cyan, magenta, yellow and black, and are said to work the CMYK color space.

Another factor is that different color devices have different color capabilities. Every color device, such as a scanner, printer, or monitor, has a range of colors that it can produce. This range of produceable colors is known as a gamut. Those skilled in the art will recognize that color display monitors can produce and display hundreds to thousands of colors. Color printers, however, have a smaller number of printable colors. Consequently, in most situations the gamut for a color display monitor exceeds the gamut for a color printer. As a result, some colors displayed on display monitor can not be produced by a color printer.

A third factor is that devices of the same type that are manufactured by different manufacturers may produce different colors, or intensities of the same colors, for the same color data. For example, color display monitors made by different manufacturers may display different colors, or intensities of the same colors, for the same RGB values. Print technologies vary drastically, and the gamut that an ink jet color printer can print may be quite different from a printer based on a different technology, such as a color laser printer. A single printer may have a fluctuating gamut depending on the paper or ink being used at the time of printing.

The way colors are sampled in different devices is another factor. Generally, the method used by monitors and scanners to capture color follows the laws of additive color mixture. Additive color mixture adds the color together to yield the result. Additive color mixture moves a color toward white, and usually results in vivid images. Printers, however, typically follows the laws of subtractive color mixture. In subtractive color mixture, color data specifies how much of a certain color to remove from white to yield the result. Consequently, a subtractive color mixture moves colors toward black or dark gray.

A color management system offers the means of transmitting color images and documents across local and wide area networks while maintaining the fidelity of the colors of the original image or document as much as possible. In January, 1993, Apple Computer, Inc. introduced ColorSync™ 1.0, a color management system in the form of a suite of utilities. The ColorSync™ Utilities are a set of routines and data structures that enable a color processing system to match colors and communicate color information between various source and destination devices. Color information is transmitted between devices via a device profile. A device profile is a data structure that describes the basic color characteristics of the device. Color information described in a device profile includes data relating to the device's color space, gamut, tonal reproduction curves, and the preferred color matching method (CMM).

Color matching means converting colors between differing gamuts. A CMM implements an algorithm that determines how to match colors. Because the ColorSync™ Utilities are designed to provide an "open system" for color management, developers can use an Apple-supplied default CMM, create their own CMM, or obtain them from companies or vendors who create CMMs.

The ColorSync™ Utilities provide applications or device drivers with several tools for matching colors between devices. The tools include a default system color profile that describes the gamut of the Apple™ RGB 13 inch monitor, a control panel interface by which users can set their system profile, a means of specifying and obtaining device profiles for other devices, a means of associating device profiles with images or documents, an Apple-supplied default CMM, a folder for storing device profiles, and an open architecture that allows developers to create or obtain a custom CMM and associate it with a device profile.

Device profiles in ColorSync™ 1.0 are memory resident, meaning a profile is read into random access memory (RAM) when information regarding data within the device profile is needed or when the device profile is being used by the ColorSync™ Utilities. In order to do this, computer systems must have a sufficient amount of RAM to hold the entire device profile while one or more application software programs are running in the computer system.

The requirement to load the entire device profile into RAM adversely affects the performance and resource requirements of the computer system. This is due to the fact that as the size of device profiles increase, more and more RAM is needed in order to access or use the device profile. As a result, an undesirable amount of RAM is consumed by the device profile, or the device profile can not be used because the computer system does not have sufficient RAM.

To address the foregoing limitations associated with prior art systems, the present invention provides a method in accordance with independent claim 1 and a system in accordance with independent claim 5. Further advantageous features, aspects and details of the invention are set forth in the dependent claims, the following description and the drawings. The claims are to be understood as a first non-limiting approach to defining the invention in general terms. In one aspect of the invention, a modular architecture of a color management system provides for multiple accesses of a device profile and manipulation of the device profile, or portions of the device profile, while maintaining the integrity of the device profile. The device profile structure is defined as a series of tagged elements that can be accessed randomly and individually.

BRIEF DESCRIPTION OF THE DRAWINGS

The novel features believed characteristic of the invention are set forth in the appended claims. The invention itself, however, as well as a preferred mode of use, and further objects and advantages thereof, will best be understood by reference to the following detailed description of an illustrative embodiment when read in conjunction with the accompanying drawings, wherein:

Figure 1 illustrates a computer system that may be used to implement the method and system for dynamically dispatching color profile data in a modular color management system according to the present invention;

Figure 2 is a block diagram depicting the architecture of a color processing system according to the present invention;

Figure 3 is a block diagram illustrating the modular architecture of a color management system according to the present invention;

Figure 4 is a pictorial representation of a color profile according to the present invention;

Figure 5 is a block diagram depicting a method and system for dispatching an element in a color profile according to the present invention; and

Figure 6 is a block diagram illustrating a method and system for dynamically dispatching color profile data in a computer system according to the present invention.

DETAILED DESCRIPTION

With reference now to the figures and in particular with reference to Figure 1, a computer system is illustrated that may be used to implement the method and system for dynamically dispatching device profile data in a color management system according to the present invention. Computer system 10 includes a computer 12, keyboard 14, a color display monitor 16 having a display screen 18, a cursor control device 20, shown here as a mouse, and a printer 22. Computer system 10 may be implemented using any suitable computer, such as a Macintosh Quadra™ computer, a product of Apple Computer, Incorporated, located in Cupertino, California. Printer 22 is a color printer, such as a Color Stylewriter Pro™ printer, also a product of Apple Computer, Incorporated.

One of the utilities available in computer system 10 is ColorSync™ 2.0, a product created by Apple Computer, Incorporated. ColorSync™ 2.0 is a color management tool that supports color information exchange, color matching, device profile management, and device calibration. Color matching is important when transferring color images and documents between color devices, such as monitors and printers. Color matching is necessary because color devices have different color capabilities, describe color in different terms and operate in different color spaces.

For example, a color display monitor 16 in computer system 10 creates and describes colors in terms of red, green and blue ("rgb") values, and is said to work in the RGB color space. The rgb values for display monitor 16 are device dependent, meaning the rgb values are particular for that monitor. Because the rgb values are device dependent, colors displayed on a different monitor will probably not match the colors displayed on display monitor 16 for the same rgb values.

Most printers create and describe colors in device dependent terms differing from monitors. Printers use cyan, magenta, yellow and black ("cmyk") values to describe colors, and are said to work in the CMYK color space. Again, because the cmyk values are device dependent, colors printed on printer 22 will probably not match colors printed on a different printer for the same cmyk values.

The need for color matching is further supported by the fact that different color devices have different color capabilities. Every color device, such as a scanner, printer, or monitor, has a range of colors that it can produce. This range of produceable colors is known as a gamut. Those skilled in the art will recognize that color display monitors can produce and display hundreds to thousands of colors. Color printers, however, have a smaller number of printable colors. Consequently, in most situations the gamut for a color display monitor exceeds the gamut for a color printer. As a result, some colors displayed on display monitor 16 can not be produced by printer 22.

Figure 2 is a block diagram depicting the architecture of a color processing system according to the present invention. Color processing system 24 includes a source device 26. Source device 26 is a device used to input a document or image into color processing system 24. For example, source device 26 can be a color display monitor, a scanner or other real or virtual devices, such as a camera or color composition routines. Block 28 represents a device driver associated with source device 26 or an application software program being used by a user to access the input document or image.

Color processing system 24 further includes a destination device 30. Examples of destination devices include a printer, a color monitor, or other real or virtual devices such as a plotter or color composition routines. Device driver 32 is associated with destination device 30.

When a user wants color processing system 24 to perform color matching on an image or document, the application software program or the device driver shown in blocks 28 or 32 calls ColorSync™ Utilities 34. The ColorSync™ Utilities 34 are a set of routines and data structures that enable color processing system 24 to match colors and communicate color information between the various source and destination devices. Color information is transmitted between devices via a device profile. A device profile is a data structure that describes the basic color characteristics of the device. Color information described in a device profile includes data relating to the device's color space, gamut, tonal reproduction curves, and the preferred color matching method (CMM). The structure and contents of a device profile will be described in greater detail with reference to Figure 4.

Still referring to Figure 2, color processing system 24 preferably has at least two device profiles. Source profile 36 is associated with source device 26. If color processing system 24 has a plurality of source devices, each source device may have its own source profile. Destination profile 38 is associated with destination device 30. If color processing system 24 has a plurality of destination devices, each destination device may have its own destination profile. Profiles can reside in files, device drivers, applications, documents, and images. Although Figure 2 depicts only one source profile and one destination profile, more than two profiles may reside in color processing system 24.

Finally, color processing system 24 includes a component manager 40 and at least one color matching method (CMM). Blocks 42, 44, 46 represent three different CMMs within color processing system 24. The Apple™ Color Matching Method included with ColorSync™ is one of the CMMs in color processing system 24. A CMM is where the conversion, or color matching, between differing color gamuts occurs.

Components are called through component manager 40. In this manner, applications and device drivers are considered "clients." A component is a piece of code that provides a defined set of services to one or more clients. A component typically provides a specific type of service to its clients. For example, a component might provide image compression or decompression. A client would call the component, provide the component with the image, and the component would then compress the image and return the compressed image to the application. Multiple components can provide the same type of service. For

example, separate components may exist that can compress an image by 20 percent, 40 percent, or 50 percent, with varying degrees of fidelity.

Component Manager 40 provides access to and management of components. An example of management of components is the tracking of the available components and routing requests to the appropriate component. The component manager 40 classifies components by three main criteria: the type of service provided, the level of service provided, and the component manufacturer. Component manager 40 uses a component type, consisting of a sequence of four characters, to identify the type of service provided by a component. CMMs have a component type of 'CMM_'. The Apple-supplied CMM component has a subcomponent type of 'appl'. Manufacturers and suppliers of other CMMs who register their CMM with Apple Computer, Inc. are assigned a subcomponent type.

Component Manager 40 provides services that allow clients to obtain run-time location of and access to functional objects by creating an interface between components and clients. A standard interface is used through which a client can communicate with all components of a given class, such as CMMs. Component Manager 40 is used to locate and communicate with components of that class. The components, in turn, provide the appropriate services to the client. A more detailed description of component manager 40 is found in Chapter 6 of *Inside Macintosh: More Toolbox Utilities* (1993) by Apple Computer, Inc.

Component manager 40 allows a single component to service multiple clients at the same time. Each client has a unique access path to the component. These access paths are called component connections. A component connection is identified by specifying a component instance. Component manager 40 provides the component instance to a client when it opens a connection to a component. The component maintains separately status information for each open connection.

Multiple clients may each open a connection to a component. Component manager 40 routes each client request to the component instance for that connection. Because a component can maintain separate storage for each connection, client requests do not interfere with each other and each client has full access to the services provided by the component.

With ColorSync™, color matching is the type of service provided to one or more clients. A separate ComponentInstance is opened for each matching session.

The set of device profiles used in the matching session is unique for each session. The color matching component should allocate private storage to store the necessary information for the instance and use Component Manager functions to manage the storage. The color matching component should free the storage when the matching session is completed by making a ComponentClose function.

As stated above, CMMs have a component of the type 'CMM_' provide color matching services. All components of type 'CMM_' share a common interface, but each component may support a unique color matching technique or take advantage of a special hardware implementation. Individual components may support additions to the defined interface, as long as they support the common routines.

The color matching process will now be described by way of example. A color image is input into color processing system 24 by a scanner (source device) and is to printed on a color printer (destination device). The user wants the color in the image printed on the color printer to be matched as close a possible to the colors scanned by the scanner. The device driver shown in block 28 associated with the scanner calls the ColorSync™ Utilities 34. The ColorSync™ Utilities 34 examines the source profile 36 associated with the scanner and the destination profile 38 associated with the color printer to determine which CMMs should be used. The ColorSync™ Utilities 34 then calls Component Manager 40 to call one or more CMMs 42, 44, 46 to perform color matching. How many CMMs are called depends upon whether or not the source profile 36 and the destination profile 38 can use the same CMM for color matching. Once color matching is completed, the image is printed on the color printer.

Referring to Figure 3, a block diagram illustrates the modular architecture of a color management system according to the present invention. The ColorSync™ Utilities 34 include at least five components, or modules. These modules are dispatcher 48, profile management 50, color space conversion 52, CMMs 42, 44, 46, and device profiles (not shown).

Dispatcher 48 provides the overall color management administrative framework and manages the interaction between the color management system, including calling device 54, component manager 40, profile management 50, color space conversion 52, device profiles (not shown), CMMs 42, 44, 46, and the actual color image or document (not shown). Dispatcher 48 is installed from the

Extensions file at system startup in the preferred embodiment. On 680x0 Macintosh™ computers, dispatcher 48 is implemented using the Toolbox A-trap mechanism. The interface defines the value for register D0 at the time a function is called. The A selector is in the low 16 bits and the parameter size is in the high 16 bits. The functions use PASCAL stack-based calling conventions. ColorSync™ 34 implements a 680x0 mainline function which internally dispatches individual calls on the selector.

The implementation of dispatcher 48 will vary for other computer systems. For example, dispatcher 48 is implemented on Power Macintosh™ computers by including a mechanism for internal dispatch of traps with a 68K register D0 selector. A function _SetTrapAddress is called with a pointer to a block of routine descriptors and selectors.

Profile management 50 provides searching, retrieval, and indexing functions for the various profiles, along with the ability to get and set individual attributes within a profile. Profile management 50 is registered at system startup with the following parameters.

```
componentType              'prof'
componentSubType           'dflt'
componentManufacturer  'appl'
```

As discussed earlier, the Apple-supplied default CMM has a subcomponent type of 'appl'. Companies or vendors who create CMMs can register their CMMs with Apple and be assigned a subcomponent type.

Search specifications are defined by the following C language code and structure.

```
typedef
pascal
Boolean (*ProfileFilterProcPtr)(Profile prof, void *refCon);

#if USESROUTINEDESCRIPTORS
typedef UniversalProcPtr ProfileFilterUPP;
#else
typedef ProfileFilterProcPtr ProfileFilterUPP;
```

```
#endif

struct ProfileSearchRecord {
        OSType          CMMType;
        OSType          profileClass;
        OSType          deviceColorSpace;
        OSType          interchangeColorSpace;
        unsigned long   deviceManufacturer;
        unsigned long   deviceModel;
        unsigned long   deviceAttributes[2];
        unsigned long   profileFlags;
        unsigned long   searchMask;
        ProfileFilterUPP    filter;
};
typedef struct ProfileSearchRecord      ProfileSearchRecord;
```

The bits of field "Mask" specify corresponding header fields which must match in the search.

```
#define     kMatchProfileCMMType        0x00000001
#define     kMatchProfileClass          0x00000002
#define     kMatchDeviceSpace           0x00000004
#define     kMatchInterchangeSpace      0x00000008
#define     kMatchManufacturer          0x00000010
#define     kMatchModel                 0x00000020
#define     kMatchAttributes            0x00000040
#define     kMatchProfileFlags          0x00000080
```

The field "Filter" is a client function which can be used to search on elements outside the header of the device profile, or to implement AND or OR search logic. If "Filter" returns TRUE for a specific device profile, then it is excluded, or filtered, from the search.

Color space conversion 52 converts color data between various independent and derived color spaces. In the preferred embodiment, the following independent and derived color spaces are supported: CIEXYZ, CIELab, CIELuv, Yxy, RGB, HLS, HSV, GRAY, CMY, CMYK, and "Hi-fi" color, which is a special purpose multi-component color space intended for commercial printing

with multiple inks. Color space conversion 52 preferably converts colors between color spaces within a color family. An example of a color family is CIEXYZ, CIELab, CIELuv and Yxy.

Figure 4 is a pictorial representation of a device profile according to the present invention. A device profile 56 contains at least three sections, a header 58, a tag table 60, and tagged element data 62. Custom profiles may also have additional, private data.

Header 60 defines a set of parameters at the beginning of device profile 56. In the preferred embodiment, header 58 has the following contents and size:

struct CM2 Header {		bytes
unsigned long	size;	0-3
OSType	CMMType;	4-7
NumVersion	profileVersion;	8-11
OSType	profileClass;	12-15
OSType	dataColorSpace;	16-19
OSType	interchangeColorSpace	20-23
CMDateTime	dateTime;	24-35
OSType	CS2profileSignature;	36-39
OSType	platform;	40-43
unsigned long	flags;	44-47
OSType	deviceManufacturer;	48-51
unsigned long	deviceModel;	52-55
unsigned long	deviceAttributes[2];	56-63
unsigned long	renderingIntent;	64-67
FixedXYZColor	white;	68-79
char	reserved[36];/*for future use*/};	80-127

The first eight bits of the profile version are the major version number and the next eight bits are for the minor version number. The platform flag indicates the primary platform/operating system of the device. The flags in bytes 44-47 are used to indicate various hints for the CMM such as distributed processing and caching options. The rendering intent may be one of four types in the preferred embodiment. The rendering intent types are perceptual, relative colorimetric, saturation and absolute colorimetric. Device attributes are attributes unique to the

particular device setup such as media type. The profile class field indicates the type of device for which the profile is intended. Three standard types are:

'scnr'	input devices such as scanners and digital cameras
'mntr'	display devices such as CRTs and LCDs
'prtr'	output devices such as printers

In addition to the three basic device profile classes, three additional color processing profiles may be used in the preferred embodiment. These profiles provide a standard implementation for use by the CMM in general color processing or for the convenience of CMMs which may use these types to store calculated transforms. These three profile classes are:

'link'	device link profiles
'spac'	color space conversion profiles
'abst'	abstract profiles

Tag table 60 provides a table of contents to tagged element data 62 in device profile 56. Tag table 60 includes a tag signature, address and size for each tagged element data 62. A tag signature acts as a pointer to a specific element in tagged element data 62. In the preferred embodiment, a tag signature is defined as a four byte hexadecimal number.

Each tagged element data 62 is a piece of color information that a CMM can use to perform color matching or correction. Examples of tagged element data 62 include gray tone reproduction curve information and the tristimulus values for red, green, and blue. By using tag table 60, tagged element data 62 can be accessed randomly and individually. In this manner, a single element 64 within tagged element data 62 can be accessed and loaded into memory, thereby providing only the information necessary to perform a particular function or process of color matching or to present device information to a user. Providing device information to a user may be accomplished via a user interface implemented by the application or device driver.

Appendix 1 provides a list of the calls or requests which can be used to access a profile in accordance with the present invention. Appendix 2 specifies the individual tags and the tag signatures that can be used in accordance with the present invention.

Referring to Figure 5, a block diagram depicts a method and system for dispatching an element in a device profile according to the present invention. A first memory 66 preferably contains a system folder 68. Within system folder 68 is a profile folder 70. Profile folder 70 preferably contains at least one device profile that may be used with ColorSync™ 2.0. In the preferred embodiment, first memory 66 is a main memory in a computer system, one example being a hard drive.

A second memory 72 contains at least a portion of a device profile 74. In the preferred embodiment, second memory 72 is any scratch or temporary memory, one example being random access memory (RAM). The portion of device profile 74 within second memory 72 preferably includes header 58, tag table 60, and possibly one element from tagged element data 62, as shown in Figure 4. When a different element is needed, caller 76 requests a new element and allocates the necessary amount of memory in second memory 72 for the new element. Caller 76 can be a CMM, device driver, application or calibration device.

The call used to request an element is stated below.

```
pascal
CMError
GetProfileElement (Profile prof, OSType tag, long *elementSize, void
        *elementData);
```

In the GetProfileElement call, "Profile prof" is an input parameter that specifies the profile to be accessed. "OSType Tag" is an input parameter that specifies the tag signature for the requested element. "ElementSize" is an input and output parameter, and specifies the size of the element to be copied. "ElementData" is an input and output parameter, and specifies the destination for copying the element. If elementSize is not NULL and elementData is NULL when a call is made, then the size of the element is returned in the *elementSize parameter. If elementSize and elementData are not NULL, then the element is copies to the *elementData parameter. Finally, If elementSize is NULL and elementData is not NULL, the entire element is copied.

The method and system of the present invention provides for other functions regarding elements in tagged element data 62. One function is the

ability to determine whether or not an element with a specific tag signature exists in the profile. The call for this function is given below.

```
pascal
CMError
ProfileElementExists(Profile prof, OSType tag, Boolean *found);
```

The parameter "Boolean *found" is an output parameter, and returns a true if the element exists. "Profile prof" and "OSType tag" are defined above.

Another function available in the preferred embodiment provides the ability to access only a part of an element for a specified tag signature. The caller of the function is responsible for allocating memory for storage of the partial element. The call for this function is stated below.

```
pascal
CMError
GetPartialProfileElement (Profile prof, OSType tag, unsigned long  offset,
                          unsigned long byteCount, void *elementData);
```

The parameter "unsigned long offset" is an input parameter that specifies where to begin the transfer of data within the element. "Unsigned long byteCount" identifies the number of bytes to transfer. "Profile prof", "OSType tag", and "*elementData" are defined above.

A user has the ability to reserve a portion of an element for a specific tag in the preferred embodiment. This function is broken down into two function calls. The first function call is given below.

```
pascal
CMError
SetProfileElementSize(Profile Prof, OSType tag, unsigned long
elementSize);
```

This call passes parameters that set the size of the element. The parameter "elementSize" is an input parameter that identifies the total byte size to be reserved for data in an element. This first call must be made prior to making the second function call. The second function call is stated below.

```
pascal
CMError
setPartialProfileElement(Profile prof, OSType tag, unsigned long offset,
        unsigned long byteCount, void *elementData);
```

The second function sets part of an element for the specified tag signature. If desired, the second function call may be repeated until the complete element has been transferred to scratch memory. These two function calls provide the color management system of the preferred embodiment with the ability to access or set portions of the data in an element.

The modular architecture of the color management system of the preferred embodiment provides for multiple accesses of the same device profile and manipulation of the device profile, or portions of the device profile, while maintaining the integrity of the device profile. Figure 6 is a block diagram illustrating a method and system for dynamically dispatching device profile data in a computer system according to the present invention. Multiple devices within a computer system may need to access the same device profile at one time. For example, a CMM 78 may need to access the device profile to obtain an element for color matching, as described with reference to Figure 5. A device driver 80 may access the same device profile to determine what is the preferred CMM in the device profile. Another access may occur because an application software program 82 wants to know whether or not a specific element exists in the device profile. Additional access to the same device profile may be done by other 84. Other 84 may be any other device or software which wants to access the device profile, such as a calibration device. Other 84 may also be a user who wishes to temporarily modify the device profile.

As discussed above with reference to Figure 3, dispatcher 86 provides the overall color management administrative framework and manages the interaction between modules within the color management system. Accesses to device profiles are routed through dispatcher 48. Device profiles are preferably stored in a profile file 88 in the main memory of a computer system. If an access to a device profile involves a modification of the device profile, a temporary copy of the modified device profile is stored in a scratch memory 90. Interaction between profile file 88 and scratch memory 90 are handled by dispatcher 86. Modification of the device profile includes adding, deleting or changing one or more elements

in the device profile, or changing information in the header or tag table in the device profile.

For example, a user may want color matching to occur when printing a document. However, for this document only, the user wants to use a rendering intent different from the one specified in the device profile. Consequently, the user needs to modify the header in the device profile. This is accomplished via a user interface in the preferred embodiment. A copy of the header with the changed rendering intent flag is stored in scratch memory 90 and referenced when printing the document. The original device profile remains unchanged in profile file 88. Consequently, any other accesses to the header in the same device profile stored in profile file 88 are not affected by the user's temporary modification of the header, since the modified header is stored in scratch memory 90. The modified header will be discarded unless the user requests the device profile be updated.

The method and system for dynamically dispatching device profile data can also be implemented in a computer network. The ability to modify the device profile or its contents while maintaining the integrity of the device profile and its contents is useful in a computer network where multiple computer systems can be using the same files or data. In this situation, one computer system preferably can not update, or make lasting changes to the device profile when other systems are accessing the device profile.

While the invention has been particularly shown and described with reference to a preferred embodiment, it will be understood by those skilled in the art that various changes in form and detail may be made therein without departing from the spirit and scope of the invention.

APPENDIX 1

Profile File and Element Access

OpenProfileFile

Open specified CMProfile file. Return a Profile reference. The profile file is opened with shared read/write permission.

```
pascal
CMError
OpenProfileFile(Profile *prof,
                const FSSpec *fileSpec);
```

Parameters

prof	(out)	Returns reference to Profile if function result is noErr. Otherwise undefined.
fileSpec	(in)	Specification for profile disk file.

Result codes

noErr	
paramErr	FSSpec is invalid

System or ColorSync™ result code if an error occurs.

UpdateProfileFile

Rewrite disk file if elements of specified Profile have been added or changed. An error will be returned if the profile file is open by another program.

```
pascal
CMError
UpdateProfileFile(Profile prof)
```

Parameters

prof	(in)	Profile reference.

Result codes

noErr

System or ColorSync™ result code if an error occurs.

CloseProfile

Close data file and free memory allocations associated with specified Profile. Changes to the file are not automatically recorded. See UpdateProfileFile(...).

```
pascal
CMError
CloseProfile(Profile prof)
```

Parameters

prof (in) Profile reference.

Result codes

noErr

System or ColorSync™ result code if an error occurs.

NewProfile

Create new empty version 2.0 Profile and associated backing file.

pascal

CMError

NewProfile(Profile *prof, const FSSpec *backingFileSpec)

Parameters

prof (out) Returns reference to new Profile if function result is noErr. Otherwise undefined.

backingFileSpec (in) Disk file specification. If NULL then a temporary Profile is created and no disk file will remain after CloseProfile(...) is called.

Result codes

noErr

dupFNErr File described by backingFileSpec exists.

System or ColorSync™ result code if an error occurs.

CopyProfile

Duplicate a Profile. Temporary changes which have been made to the source Profile are included in target CMProfile disk file.

pascal

CMError

CopyProfile(Profile *targetProf,
const FSSpec *backingFileSpec,
Profile prof);

Parameters

targetProf (out) Returns reference to duplicate Profile if function result is noErr. Otherwise undefined.

backingFileSpec (in) Disk file specification for CMProfile copy.

prof (in) Profile to be duplicated.

Result codes

noErr

paramErr Invalid backingFileSpec

dupFNErr File described by backingFileSpec exists.

System or ColorSync™ result code if an error occurs.

ValidateProfile

Test whether Profile contains the minimum set of elements. This function is dispatched to the CMM specified by the Profile CMMType header field. Only the existence of Profile elements is checked. Eement content and size are not.

Third-party CMMs should call the default CMM using Component Manager functions to assure the Profile contains the minimum default elements.

```
pascal
CMError
ValidateProfile(Profile prof, Boolean *valid,
                Boolean *preferredCMMnotfound);
```

Parameters

prof	(in)	Profile.
valid	(in/out)	Returns true if validation test is successful. Otherwise returns false.
preferredCMMnotfound	(out)	Returns true if the CMM corresponding to profile was not available and the default CMM was used. Otherwise returns false.

Result codes

noErr

System or ColorSync™ result code if an error occurs.

ProfileElementExists

Test whether an element with specified tag signature is present in Profile.

```
pascal
CMError
ProfileElementExists(Profile prof, OSType tag,
                     Boolean *found);
```

Parameters

prof	(in)	Profile.
tag	(in)	Element signature.
found	(out)	Returns true if element exists. Otherwise false.

Result codes

noErr

System or ColorSync™ result code if an error occurs.

CountProfileElements

Return one-based count of elements contained in Profile. Tags which are references are counted as elements.

pascal

CMError

CountProfileElements(Profile prof, unsigned long *elementCount)

Parameters

prof	(in)	Profile.
elementCount	(out)	Returns element count if function result is noErr. Otherwise undefined.

Result codes

noErr

System or ColorSync™ result code if an error occurs.

GetProfileElement

Get element data for specified tag. The caller is responsible for allocating storage.

If elementSize is non-NULL and elementData is NULL then the size of the element is returned in the *elementSize parameter.

If elementData is non-NULL then the element data is copied to the *elementData parameter. In this case the *elementSize parameter should contain the number of bytes to be copied. If elementSize is NULL then the entire element is copied.

If both elementSize and elementData are NULL then this function does nothing.

This function does not copy more bytes than necessary to transfer the element.

Upon return, if elementSize is non-NULL then the *elementSize parameter contains the size of the element (which may be greater than the number of bytes actually copied).

pascal

CMError

GetProfileElement(Profile prof, OSType tag,
unsigned long *elementSize, void *elementData);

Parameters

prof	(in)	Profile.
tag	(in)	Element tag signature.

elementSize (in/out) Returns size if elementData is NULL. Otherwise specifies size of data to be copied.

elementData (in/out) Destination for copy of element data.

Result codes

noErr

System or ColorSync™ result code if an error occurs.

GetProfileHeader

Get copy of header for the specified Profile.

This function uses a union structure with variants for version 1.0 and 2.0 ColorSync™ profile headers. The 32-bit version value is located at the same offset in either header. The caller can inspect the version and interpret the remaining header fields accordingly.

```
union AppleProfileHeader {
    CMHeader    cs1;
    CM2Header   cs2;
};
typedef union AppleProfileHeader  AppleProfileHeader;
pascal
CMError
GetProfileHeader(Profile prof, AppleProfileHeader *header);
```

Parameters

prof	(in)	Profile.
header	(out)	Copy of header.

Result codes

noErr

System or ColorSync™ result code if an error occurs.

GetPartialProfileElement

Get part of element data for specified tag. The caller is responsible for allocating storage.

```
pascal
CMError
GetPartialProfileElement(Profile prof, OSType tag,
                         unsigned long offset,
                         unsigned long *byteCount,
                         void *elementData);
```

Parameters

prof	(in)	Profile.
tag	(in)	Element tag signature.
offset	(in)	Offset within element data to begin transfer.
byteCount	(in/out)	Specifies number of bytes to transfer. Returns number of bytes actually transferred.
elementData	(in)	Buffer to receive element data.

Result codes

noErr

System or ColorSync™ result code if an error occurs.

SetProfileElementSize

Reserve size of element data for specified tag. This function must be used before calling SetPartialProfileElement(...).

```
pascal
CMError
SetProfileElementSize(Profile prof, OSType tag,
                      unsigned long elementSize);
```

Parameters

prof	(in)	Profile.
tag	(in)	Element tag signature.
elementSize	(in)	Total byte size to be reserved for element data .

Result codes

noErr

System or ColorSync™ result code if an error occurs.

SetPartialProfileElement

Set part of element data for specified tag.

```
pascal
CMError
SetPartialProfileElement(Profile prof, OSType tag,
                         unsigned long offset,
                         unsigned long byteCount,
                         void *elementData);
```

Parameters

prof	(in)	Profile.
tag	(in)	Element tag signature.

offset	(in)	Offset within element data to begin transfer.
byteCount	(in)	Number of bytes to transfer.
elementData	(in)	Buffer which is source of element data.

Result codes

noErr

System or ColorSync™ result code if an error occurs.

GetIndProfileElementInfo

Obtain element tag and data size by index in the range of 1..elementCount (as returned by CountProfileElements(...)).

```
pascal
CMError
GetIndProfileElementInfo(Profile prof, unsigned long index,
                         OSType *tag, unsigned long *elementSize,
                         Boolean *refs);
```

Parameters

prof	(in)	Profile.
index	(in)	One-based element index.
tag	(out)	Returns signature of element if function result is noErr. Otherwise undefined.
elementSize	(out)	Returns size of element data if function result is noErr. Otherwise undefined.
refs	(out)	Returns true if more than one tag references element data associated with this tag.

Result codes

noErr

System or ColorSync™ result code if an error occurs.

GetIndProfileElement

Get data for element at specified index. Caller is responsible for allocating storage.

If elementSize is non-NULL and elementData is NULL then the size of the element is returned in the *elementSize parameter.

If elementData is non-NULL then the element data is copied to the *elementData parameter. In this case the *elementSize parameter should contain the number of bytes to be copied. If elementSize is NULL then the entire element is copied.

If both elementSize and elementData are NULL then this function does nothing.

This function does not copy more bytes than necessary to transfer the element.

Upon return, if elementSize is non-NULL then the *elementSize parameter contains the size of the element (which may be greater than the number of bytes actually copied).

```
pascal
CMError
GetIndProfileElement(Profile prof, unsigned long index,
                     unsigned long *elementSize, void *elementData);
```

Parameters

prof	(in)	Profile.
index	(in)	One-based element index.
elementSize	(in/out)	Returns size if element Data is NULL. Otherwise specifies size of data to be copied.
elementData	(in/out)	Destination for copy of element data.

Result codes

noErr

System or ColorSync™ result code if an error occurs.

SetProfileElement

Set element data for specified tag. If an element with the specified tag is already present in the Profile, the existing element data is replaced.

```
pascal
CMError
SetProfileElement(Profile prof, OSType tag,
                  unsigned long elementSize, void *elementData);
```

Parameters

prof	(in)	Profile.
tag	(in)	Element signature.
elementSize	(in)	Size of element data in bytes.
elementData	(in)	Pointer to element data.

Result codes

noErr

System or ColorSync™ result code if an error occurs.

SetProfileHeader

Set header for the specified Profile.

```
pascal
```

CMError

SetProfileHeader(Profile prof, const AppleProfileHeader *header);

Parameters

prof	(in)	Profile.
header	(in)	New header.

Result codes

noErr

System or ColorSync™ result code if an error occurs.

SetProfileElementReference

Add tag to Profile which references data corresponding to a previously set element.

After successful completion of this function there will be more than one tag corresponding to a single piece of data. All of these tags are "equal citizens".

If SetProfileElement(...) is subsequently called for one of the common tags then the new element data will be "appended" to the Profile rather than replacing the existing data.

pascal

CMError

SetProfileElementReference(Profile prof, OSType elementTag,
OSType referenceTag)

Parameters

prof	(in)	Profile.
elementTag	(in)	Original element signature.
referenceTag	(in)	New tag signature refering to original data.

Result codes

noErr

System or ColorSync™ result code if an error occurs.

RemoveProfileElement

Remove element with specified tag signature from Profile.

pascal

CMError

RemoveProfileElement(Profile prof, OSType tag);

Parameters

prof	(in)	Profile.

tag (in) Element signature.

Result codes

noErr

System or ColorSync™ result code if an error occurs.

APPENDIX 2

Tag Descriptions

This section specifies the individual tags used to create all possible portable profiles in a Profile. The appropriate tag typing is indicated with each individual tag description.

Tag Name	General Description
AToB0Tag	Multidimensional transformation structure
AToB1Tag	Multidimensional transformation structure
AToB2Tag	Multidimensional transformation structure
blueColorantTag	Relative XYZ values of blue phosphor
blueTRCTag	Blue channel tone reproduction curve
BToA0Tag	Multidimensional transformation structure
BToA1Tag	Multidimensional transformation structure
BToA2Tag	Multidimensional transformation structure
calibrationDateTimeTag	Profile calibration date and time
charTargetTag	Characterization target such as IT8/7.2
copyrightTag	7 bit ASCII profile copyright information
deviceMfgDescTag	displayable description of device manufacturer
deviceModelDescTag	displayable description of device model
gamutTag	Out of Gamut : 8 or 16 bit data
grayTRCTag	Gray tone reproduction curve
greenColorantTag	Relative XYZ values of green phosphor
greenTRCTag	Green channel tone reproduction curve
luminanceTag	Absolute luminance for emissive device
measurementTag	Alternative measurement specification information
mediaBlackPointTag	Media XYZ black point
mediaWhitePointTag	Media XYZ white point
namedColorTag	Dictionary for converting between named colors and interchange or device color spaces
preview0Tag	Preview transformation : 8 or 16 bit data
preview1Tag	Preview transformation : 8 or 16 bit data
preview2Tag	Preview transformation : 8 or 16 bit data

profileDescriptionTag	
profileSequenceDescTag	
ps2CRD0Tag	PostScript Level 2 color rendering dictionary: perceptual
ps2CRD1Tag	PostScript Level 2 color rendering dictionary: colorimetric
ps2CRD2Tag	PostScript Level 2 color rendering dictionary: saturation
ps2CSATag	PostScript Level 2 color space array
ps2RenderingIntentTag	PostScript Level 2 Rendering Intent
redColorantTag	Relative XYZ values of red phosphor
redTRCTag	Red channel tone reproduction curve
screeningDescTag	Screening attributes description
screeningTag	Screening attributes such as frequency, angle and spot
technologyTag	Device technology information such as LCD, CRT, Dye Sublimation, etc.
ucrbgTag	Under color removal curve
viewingCondDescTag	Specifies viewing condition description
viewingConditionsTag	Specifies viewing condition parameters

2.1 AToB0Tag
Tag Types : lut8Type xor lut16Type
Tag Signature : 'A2B0' 0x41324230
Device to PCS : 8 bit or 16 bit data.

2.2 AToB1Tag
Tag Types : lut8Type xor lut16Type
Tag Signature : 'A2B1' 0x41324231
Device to PCS : 8 bit or 16 bit data.

2.3 AToB2Tag
Tag Types : lut8Type xor lut16Type
Tag Signature : 'A2B2' 0x41324232
Device to PCS : 8 bit or 16 bit data.

2.4 blueColorantTag
Tag Type : XYZType
Tag Signature : 'bXYZ' 0x6258595A

The relative XYZ values of blue phosphor or colorant.

2.5 blueTRCTag
Tag Type : curveType
Tag Signature : 'bTRC' 0x62545243
Blue channel tone reproduction curve. The first element represents no colorant (white) or phosphors (black) and the last element represents 100 percent colorant (blue) or 100 percent phosphor (blue).

The count value specifies the number of entries in the curve table except as follows:

when count is 0, then a linear response (slope equal to 1.0) is assumed,
when count is 1, then the data entry is interpreted as a simple gamma value
(ranging from 0 to 8 in fixed unsigned 8.8 format), and
when count is 2, the entries are intepreted as the beginning and end points of a line.

Gamma is interpreted canonically and NOT as an inverse.

2.6 BToA0Tag
Tag Types : lut8Type xor lut16Type
Tag Signature : 'B2A0' 0x42324130
PCS to Device space : 8 bit or 16 bit data.

2.7 BToA1Tag
Tag Types : lut8Type xor lut16Type
Tag Signature : 'B2A1' 0x42324131
PCS to Device space : 8 bit or 16 bit data.

2.8 BToA2Tag
Tag Types : lut8Type xor lut16Type
Tag Signature : 'B2A2' 0x42324132
PCS to Device space : 8 bit or 16 bit data.

2.9 calibrationDateTimeTag
Tag Type : dateTimeType
Tag Signature : 'calt' 0x63616C74
Profile calibration date and time. Initially, this tag matches the contents of the creationDateTime header flag. This allows applications and utilities to verify if this profile matches a vendor's profile and how recently calibration has been performed.

2.10 charTargetTag
Tag Type : textType
Tag Signature : 'targ' 0x74617267
This tag contains the measurement data for a characterization target such as IT8.7/2. This tag is provided so that distributed utilities can create transforms "on the fly" or check the current performance against the original device performance. The tag embeds the exact data file format defined in the ANSI or ISO standard which is applicable to the device being characterized. Examples are the data formats described in ANSI IT8.7/1-1993 section 4.10, ANSI IT8.7/2-1993 section 4.10 and ANSI IT8.7/3 section 4.10. Each of these file formats contains an identifying character string as the first few bytes of the format, allowing an external parser to determine which data file format is being used. This provides the facilities to include a wide range of targets using a variety of measurement specifications in a standard manner.

2.11 copyrightTag
Tag Type : textType
Tag Signature : 'cprt' 0x63707274
This tag contains the 7 bit ASCII text copyright information for the profile.

2.12 deviceMfgDescTag
Tag Type : textDescriptionType
Tag Signature : 'dmnd' 0x646D6E64
Structure containing invariant and localizable versions of the device manufacturer for display.

2.13 deviceModelDescTag
Tag Type : textDescriptionType
Tag Signature : 'dmdd' 0x646D6464

Structure containing invariant and localizable versions of the device manufacturer for display.

2.14 gamutTag
Tag Types : lut8Type xor lut16Type
Tag Signature : 'gamt' 0x67616D74
Out of Gamut tag : 8 bit or 16 bit data.

2.15 grayTRCTag
Tag Type : curveType
Tag Signature : 'kTRC' 0x6B545243
Gray tone reproduction curve. The tone reproduction curve provides the necessary information to convert between a single device channel and the CIEXYZ encoding of the profile connection space. The first element represents no colorant (white) or phosphors (black) and the last element represents 100 percent colorant (black) or 100 percent phosphor (white).

The count value specifies the number of entries in the curve table except as follows:

when count is 0, then a linear response (slope equal to 1.0) is assumed,
when count is 1, then the data entry is interpreted as a simple gamma value
(ranging from 0 to 8 in fixed unsigned 8.8 format), and
when count is 2, the entries are intepreted as the beginning and end points of a line.

Gamma is interpreted canonically and NOT as an inverse.

2.16 greenColorantTag
Tag Type : XYZType
Tag Signature : 'gXYZ' 0x6758595A
Relative XYZ values of green phosphor or colorant.

2.17 greenTRCTag
Tag Type : curveType
Tag Signature : 'gTRC' 0x67545243

Green channel tone reproduction curve. The first element represents no colorant (white) or phosphors (black) and the last element represents 100 percent colorant (green) or 100 percent phosphor (green).

The count value specifies the number of entries in the curve table except as follows:

when count is 0, then a linear response (slope equal to 1.0) is assumed,

when count is 1, then the data entry is interpreted as a simple gamma value (ranging from 0 to 8 in fixed unsigned 8.8 format), and

when count is 2, the entries are intepreted as the beginning and end points of a line.

Gamma is interpreted canonically and NOT as an inverse.

2.18 luminanceTag
Tag Types : XYZType
Tag Signature : 'lumi' 0x6C756D69
Absolute tristimulus luminance of devices in candelas per meter squared.

2.19 measurementTag
Tag Type : measurementType
Tag Signature : 'meas' 0x6D656173
Alternative measurement specification such as a D65 illuminant instead of the default D50.

2.20 mediaBlackPointTag
Tag Type : XYZType
Tag Signature : 'bkpt' 0x626b7074
This tag specifies the media black point and is used for generating absolute colorimetry. It is referenced to the profile connection space. If this tag is not present, it is assumed to be (0,0,0).

2.21 mediaWhitePointTag
Tag Type : XYZType
Tag Signature : 'wtpt' 0x77747074

This tag specifies the media white point and is used for generating absolute colorimetry. It is referenced to the profile connection space. If this tag is not present, it is assumed to be the same as the illuminant in the header.

2.22 namedColorTag
Tag Type : namedColorType
Tag Signature : 'ncol' 0x6E636F6C
Named color reference transformation for converting between named color sets and the profile connection space or device color spaces.

2.23 preview0Tag
Tag Types : lut8Type xor lut16Type
Tag Signature : 'pre0' 0x70726530
Preview transformation from PCS to device space and back to the PCS : 8 bit or 16 bit data.

2.24 preview1Tag
Tag Types : lut8Type xor lut16Type
Tag Signature : 'pre1' 0x70726531
Preview transformation from the PCS to device space and back to the PCS : 8 bit or 16 bit data.

2.25 preview2Tag
Tag Types : lut8Type xor lut16Type
Tag Signature : 'pre2' 0x70726532
Preview transformation from PCS to device space and back to the PCS : 8 bit or 16 bit data.

2.26 profileDescriptionTag
Tag Type : textDescriptionType
Tag Signature : 'desc' 0x64657363
Structure containing invariant and localizable versions of the profile description for display. This invariant description has no fixed relationship to the actual profile disk file name.

2.27 profileSequenceDescTag
Tag Type : profileSequenceDescType
Tag Signature : 'pseq' 0x70736571
Structure containing a description of the profile sequence from source to destination, typically used with the devicelink profile.

2.28 redColorantTag
Tag Type : XYZType
Tag Signature : 'rXYZ' 0x7258595A
Relative XYZ values of red phosphor or colorant.

2.29 redTRCTag
Tag Type : curveType
Tag Signature : 'rTRC' 0x72545243
Red channel tone reproduction curve. The first element represents no colorant (white) or phosphors (black) and the last element represents 100 percent colorant (red) or 100 percent phosphor (red).

The count value specifies the number of entries in the curve table except as follows:
when count is 0, then a linear response (slope equal to 1.0) is assumed,
when count is 1, then the data entry is interpreted as a simple gamma value
(ranging from 0 to 8 in fixed unsigned 8.8 format), and
when count is 2, the entries are intepreted as the beginning and end points of a line.
Gamma is interpreted canonically and NOT as an inverse.

2.30 screeningDescTag
Tag Type : textDescriptionType
Tag Signature : 'scrd' 0x73637264
Structure containing invariant and localizable versions of the screening conditions.

2.31 screeningTag
Tag Type : screeningType
Tag Signature : 'scrn' 0x7363726E
This tag contains screening information for a variable number of channels.

2.32 technologyTag
Tag Type : signatureType
Tag Signature : 'tech' 0x74656368
Device technology information such as CRT, Dye Sublimation, etc.

The encoding is such that :

Technology	signature	hex signature
Film Scanner	'fscn'	0x6673636E
Reflective Scanner	'rscn'	0x7273636E
Ink Jet Printer	'ijet'	0x696A6574
Thermal Wax Printer	'twax'	0x74776178
Electrophotographic Printer	'epho'	0x6570686F
Electrostatic Printer	'esta'	0x65737461
Dye Sublimation Printer	'dsub'	0x64737562
Photographic Paper Printer	'rpho'	0x7270686F
Film Writer	'fprn'	0x6670726E
Video Monitor	'vidm'	0x76696460
Video Camera	'vidc'	0x76696463
Projection Television	'pjtv'	0x706A7476
Cathode Ray Tube Display	'CRT '	0x43525420
Passive Matrix Display	'PMD '	0x504D4420
Active Matrix Display	'AMD '	0x414D4420
Photo CD	'KPCD'	0x4B504344
PhotoImageSetter	'imgs'	0x696D6773
Gravure	'grav'	0x67726176
Offset Lithography	'offs'	0x6F666673
Silkscreen	'silk'	0x73696C6B
Flexography	'flex'	0x666C6578

2.33 ucrbgTag
Tag Type : ucrbgType
Tag Signature : 'bfd ' 0x62666420
Under color removal and black generation specification. This tag contains curve information for both under color removal and black generation in addition to a general description.

2.34 viewingCondDescTag
Tag Type : textDescriptionType
Tag Signature : 'vued' 0x76756564
Structure containing invariant and localizable versions of the viewing conditions.

2.35 viewingConditionsTag
Tag Type : viewingConditionsType
Tag Signature : 'view' 0x76696577
Viewing conditions parameters.

CLAIMS

1. A method for dynamically dispatching data from at least one device profile, comprising the steps of:

receiving a request for data from a device profile from at least one client; and

obtaining the requested data from the device profile for each client.

2. The method of Claim 1, wherein said method comprises the further step of storing a copy of the device profile in a buffer if a call from a client requires the device profile to be modified.

3. The method of Claim 2, wherein the buffer is a random access memory (RAM).

4. The method of one of Claims 1 to 3, wherein the step of obtaining the requested data from the device profile for each client comprises the steps of:

obtaining a tag signature from each request; and
obtaining the requested data based upon the tag signature.

5. A system having at least one client and at least one device profile, said system comprising a dispatcher to receive a request for data from a device profile from at least one client, wherein the request comprises a tag signature, and wherein upon receipt of the request the dispatcher obtains the data from the device profile based upon the tag signature.

6. The system of Claim 5, further comprising a memory for permanently storing all device profiles in a single location.

7. The system of Claim 6, wherein the memory is a read only memory (ROM).

8. The system of Claim 5 or 6 or 7, further comprising a buffer for storing a copy of the device profile.

9. The system of Claim 8, wherein the buffer is a random access memory (RAM).

10. A system for dynamically dispatching data from at least one device profile, comprising the steps of:

means for receiving a request for data from a device profile from at least one client; and

means for obtaining the requested data from the device profile for each client.

11. The system of Claim 10, further comprising a buffer for storing a copy of a device profile if a request from a client requires the device profile to be modified.

12. The system of Claim 11, wherein the buffer is a random access memory (RAM).

13. The system of Claim 11 or 12, further comprising means for using the copied device profile for any additional requests for data from the copied device profile from the client who requested the device profile be modified.

14. The system of one of Claims 10 to 13 , wherein the means for obtaining the requested data from the device profile for each client comprises:

means for obtaining a tag signature from each request; and

means for obtaining the requested data based upon the tag signature.

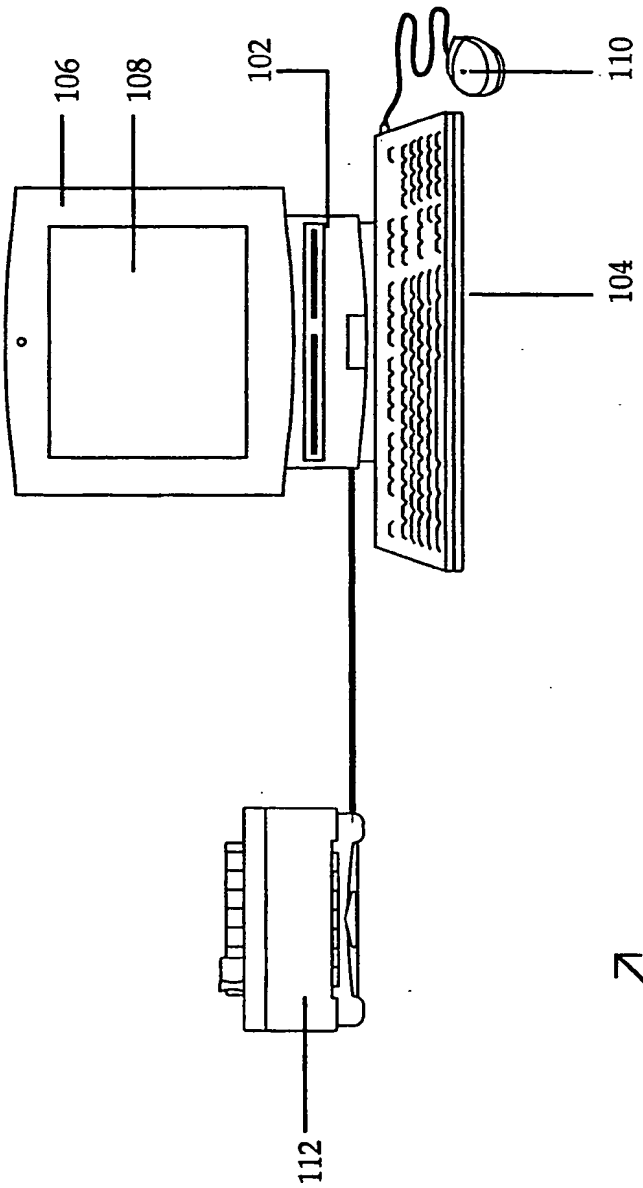


Fig. 1

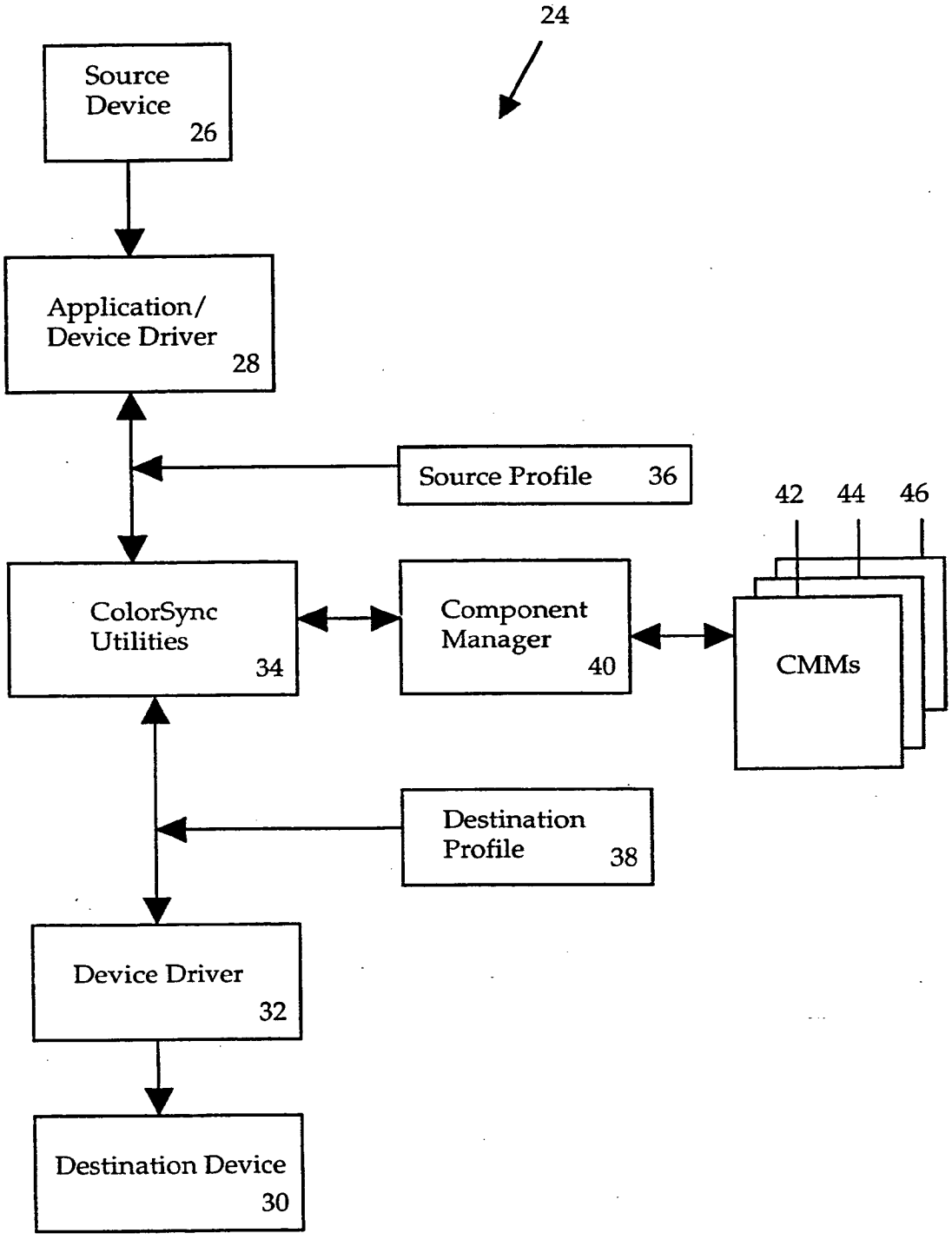


Fig. 2

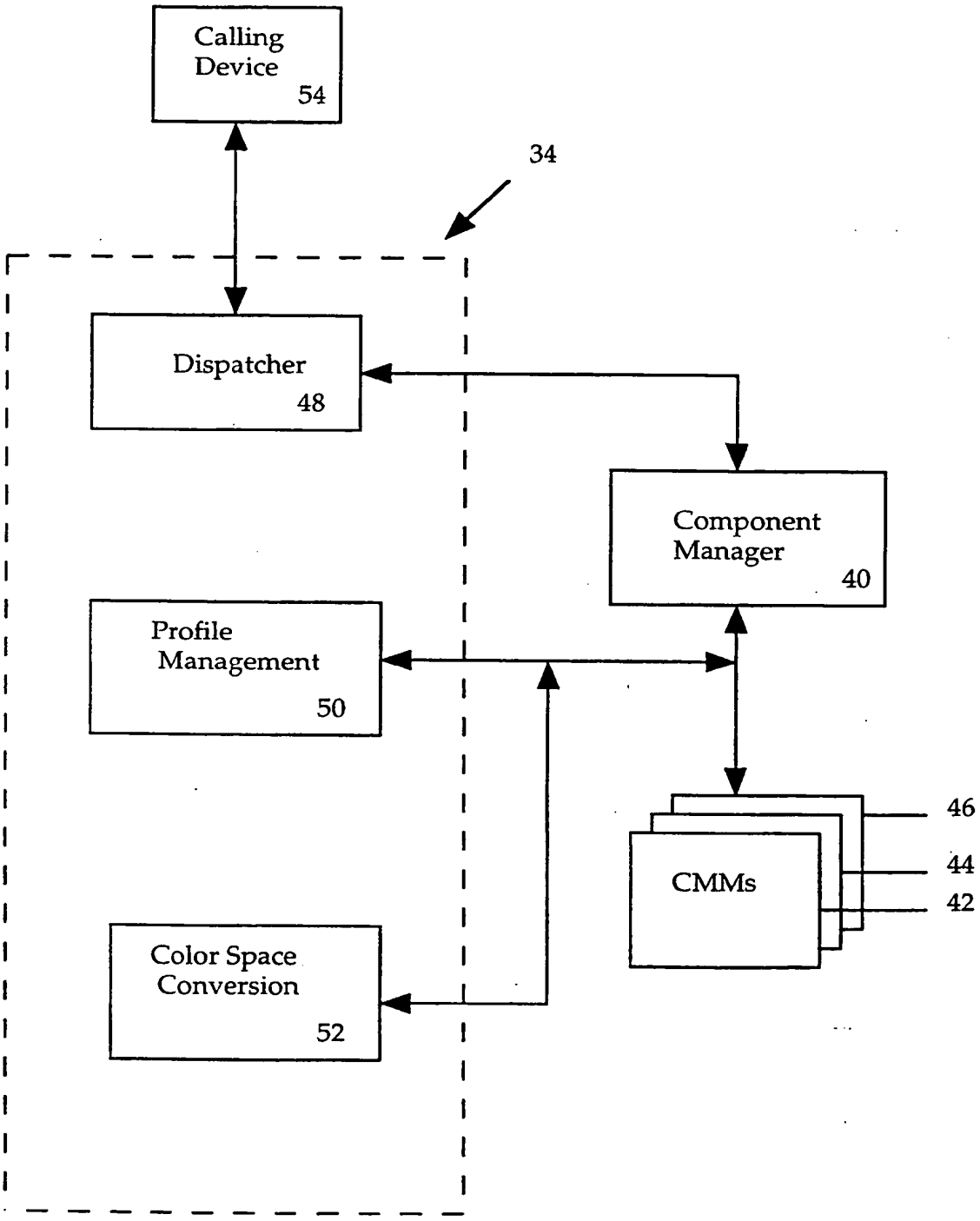


Fig. 3

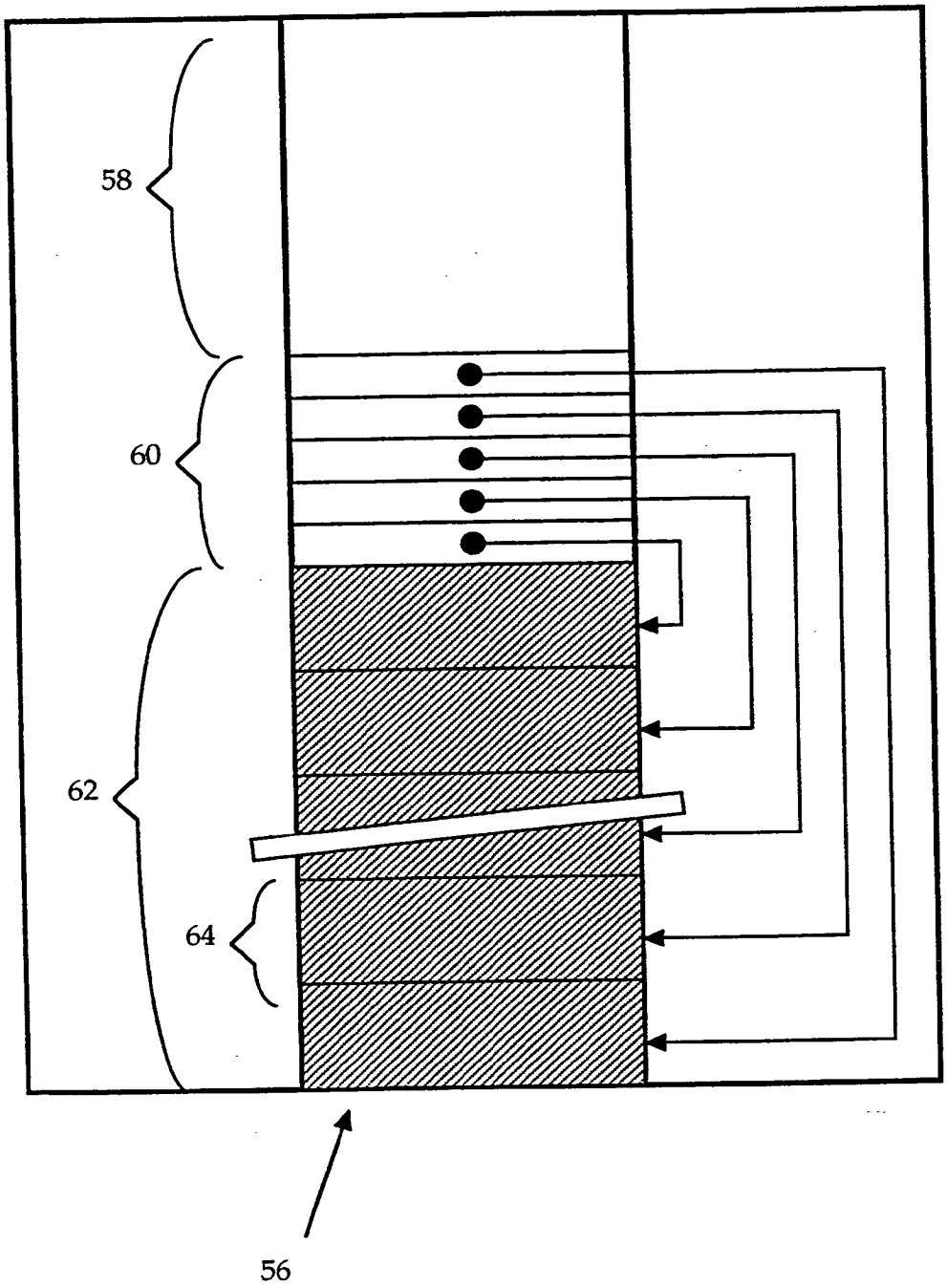


Fig. 4

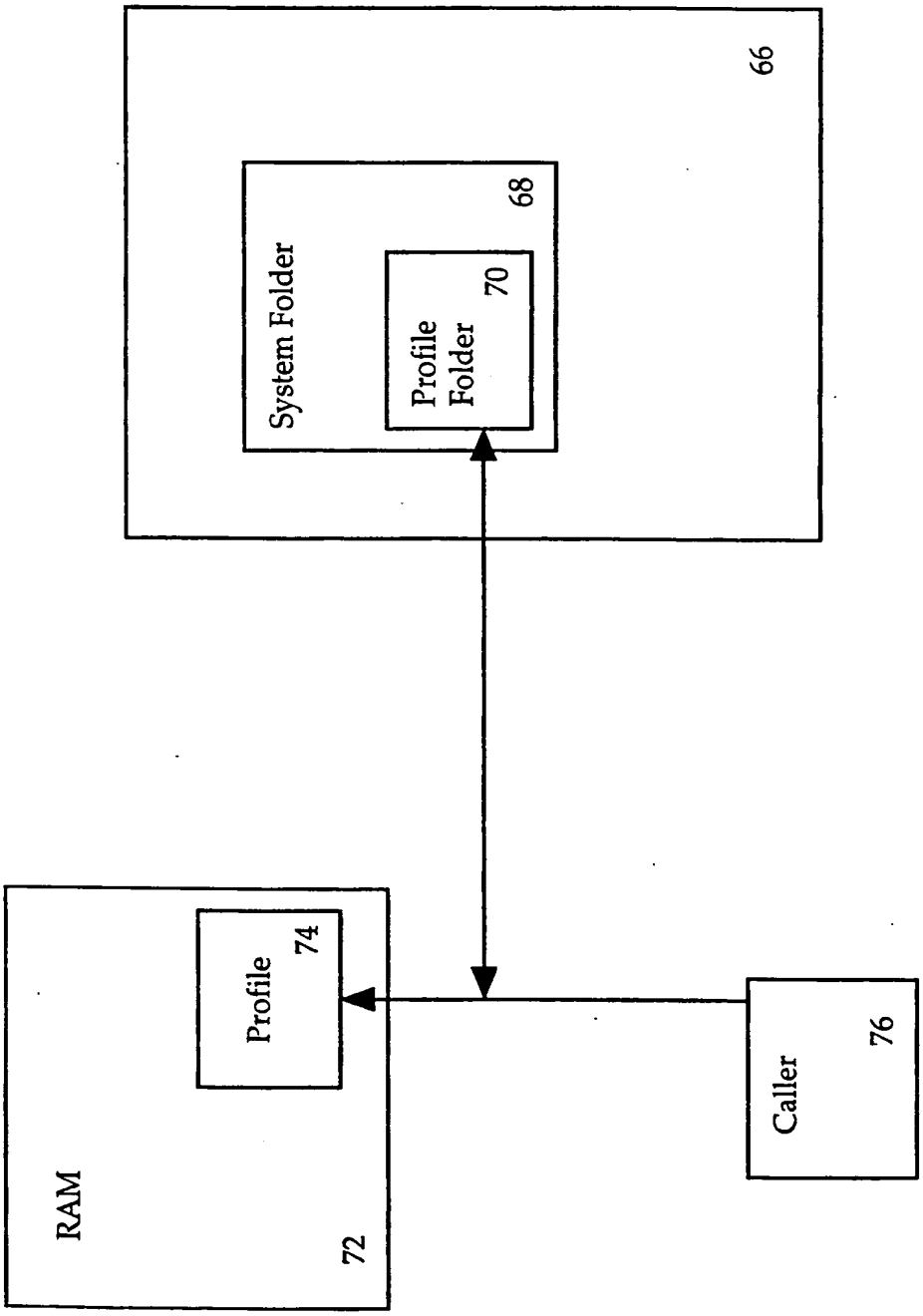


Fig. 5

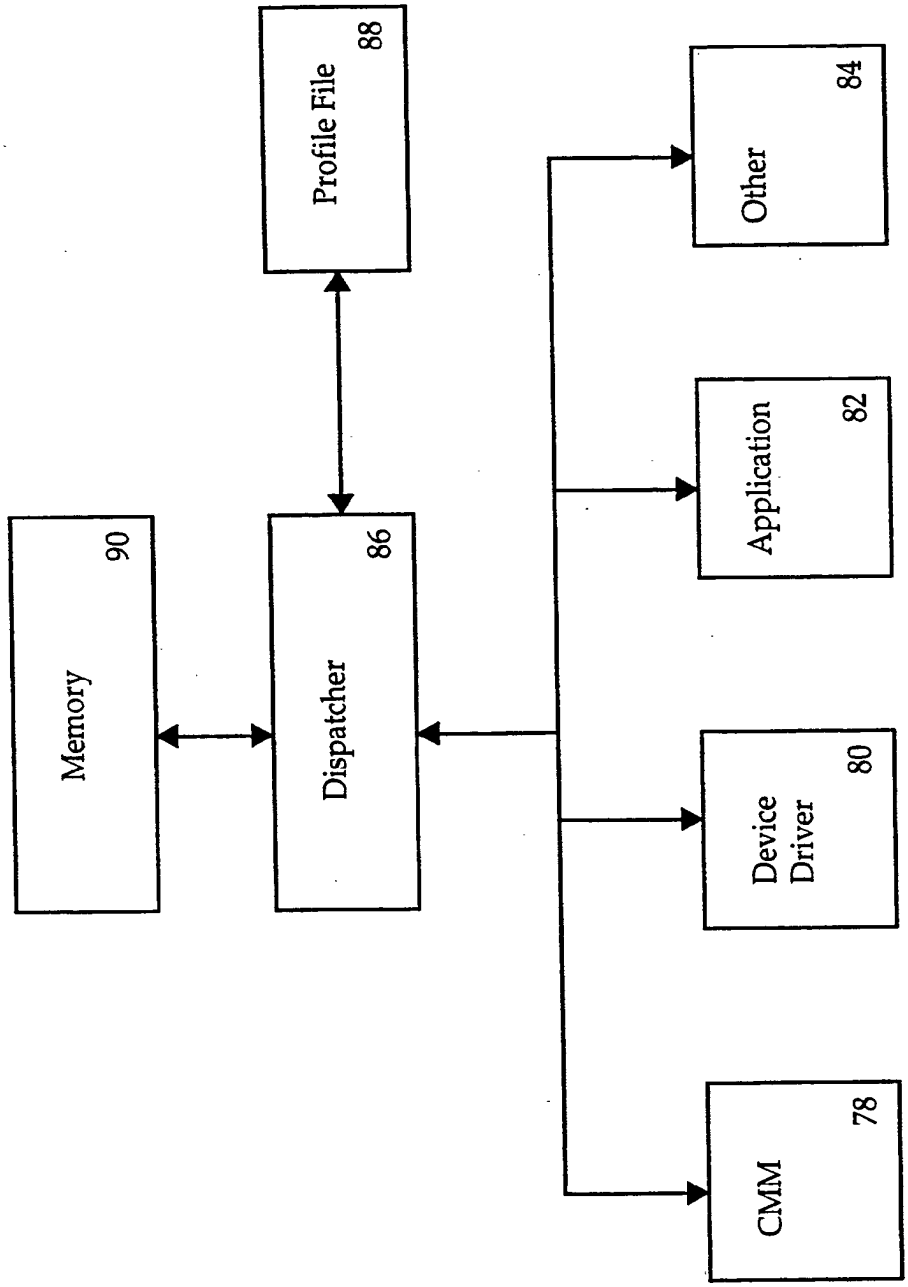


Fig. 6

INTERNATIONAL SEARCH REPORT

Inter nal Application No
PCT/US 95/08257

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 G09G5/02

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 6 G09G H04N

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	IS&T 46TH ANNUAL CONFERENCE, May 1993 SPRINGFIELD, VA, USA, pages 139-141, W. J. DONOVAN ET AL. 'Generic Architecture for Color Data Interchange' ---	
A,P	BYTE, vol. 20, no. 1, January 1995 ST PETERBOROUGH US, pages 93-100, XP 000491951 M. SUGIHARA 'Consistent Color' ---	
P,A	WO,A,94 30003 (TALIGENT INC.) 22 December 1994 -----	

☐ Further documents are listed in the continuation of box C.

☒ Patent family members are listed in annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
13 November 1995	22. 11. 95
Name and mailing address of the ISA European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax (+31-70) 340-3016	Authorized officer Farricella, L

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Information on patent family members

International Application No
PCT/US 95/08257

Patent document cited in search report	Publication date	Patent family member(s)		Publication date
WO-A-9430003	22-12-94	AU-B-	6023694	03-01-95
		CA-A-	2144872	22-12-94

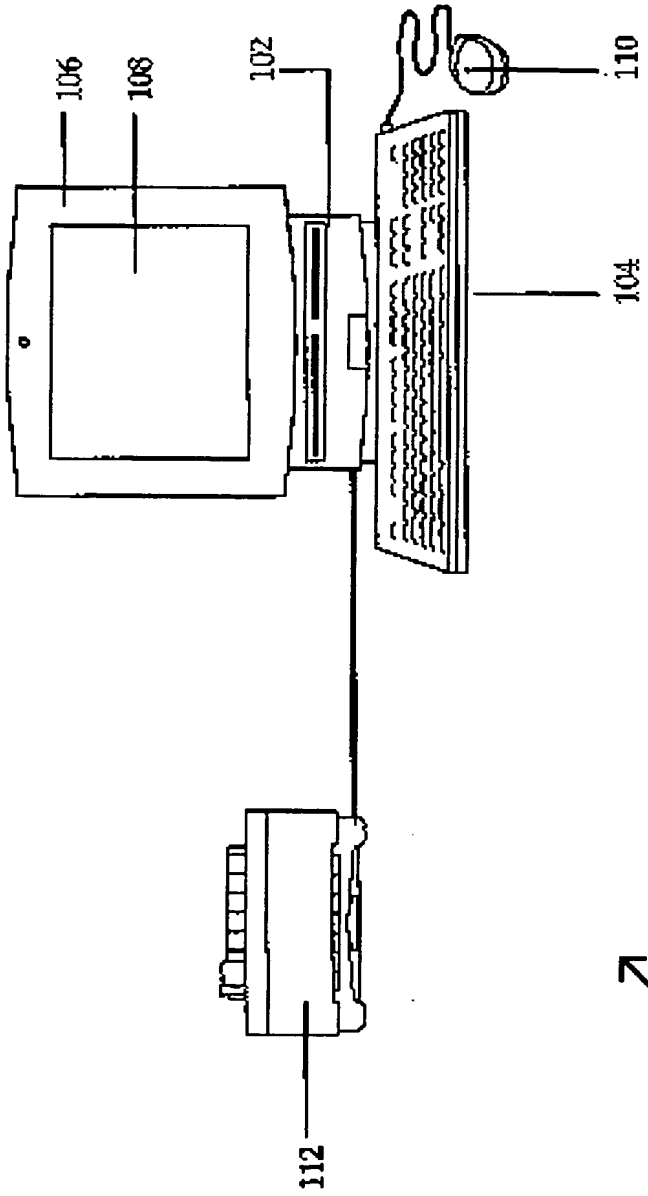


Fig. 1

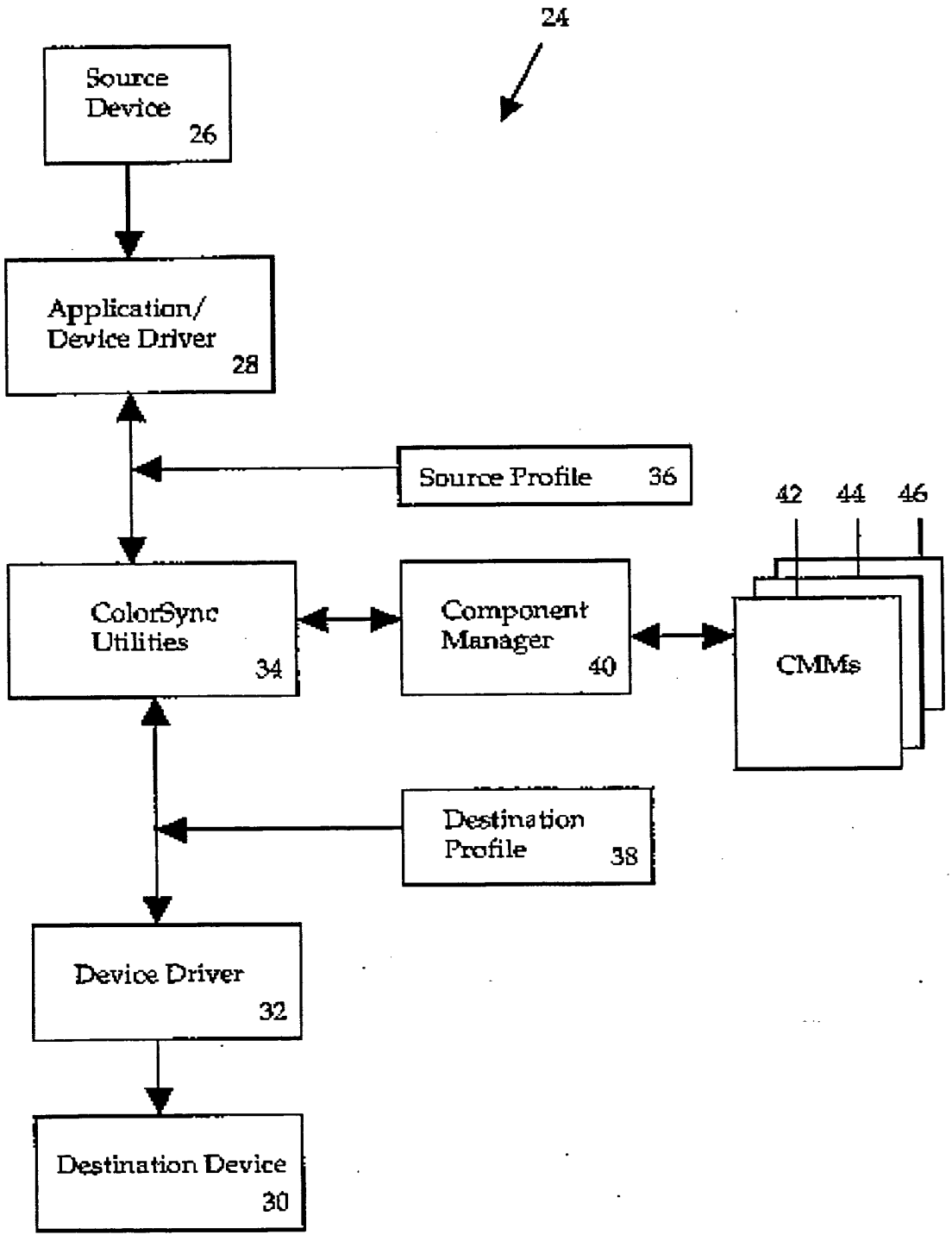


Fig. 2

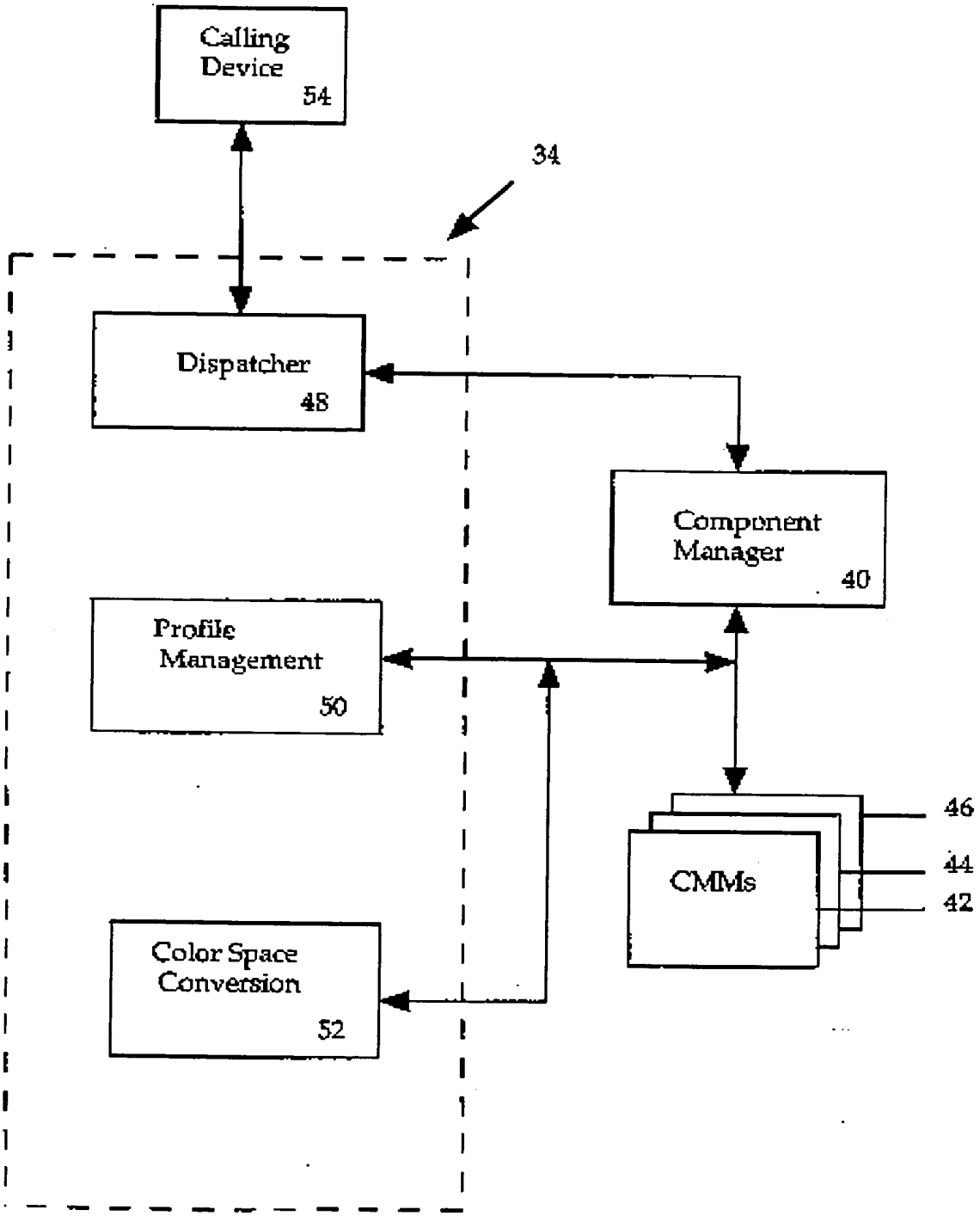


Fig. 3

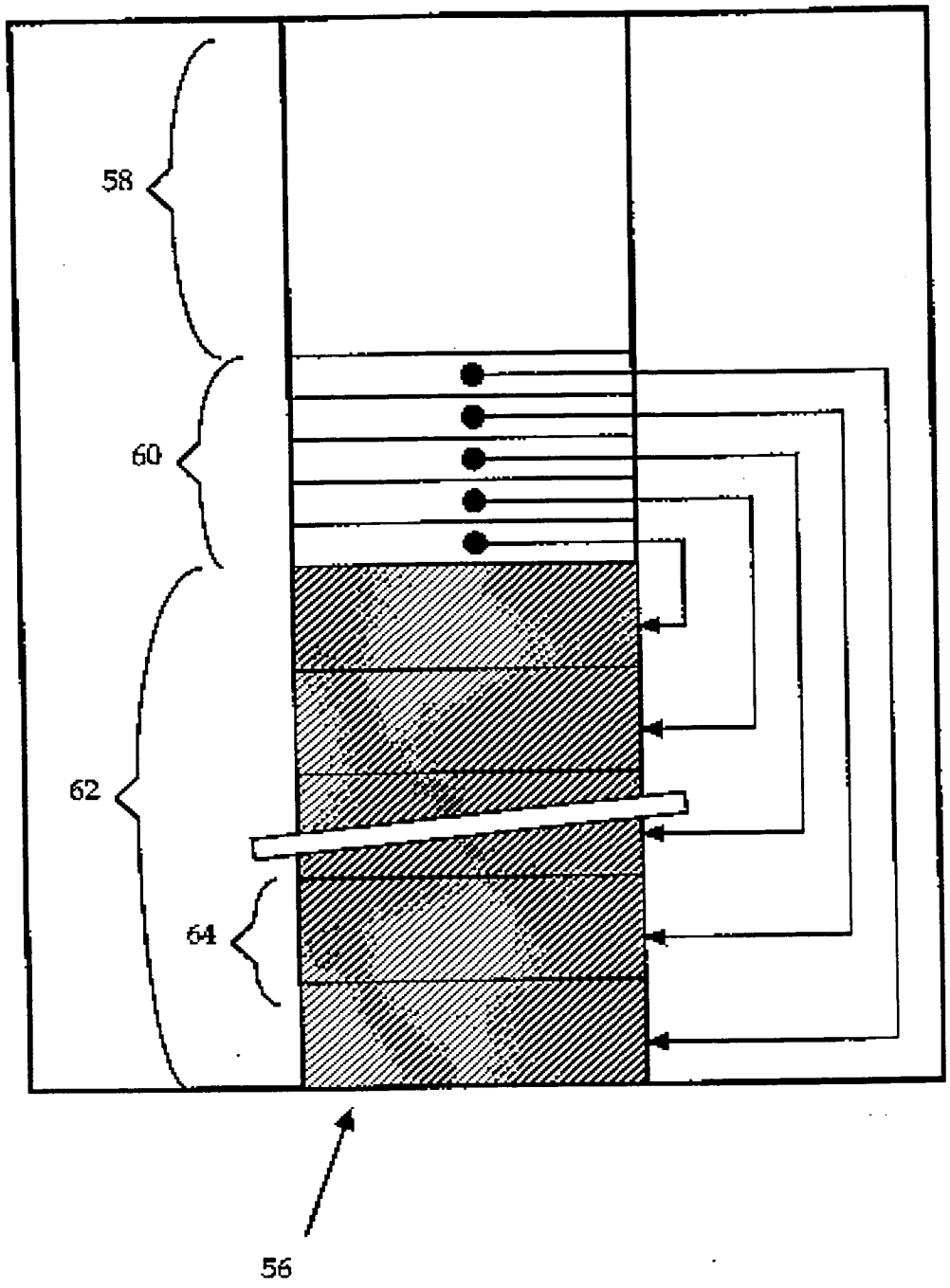


Fig. 4

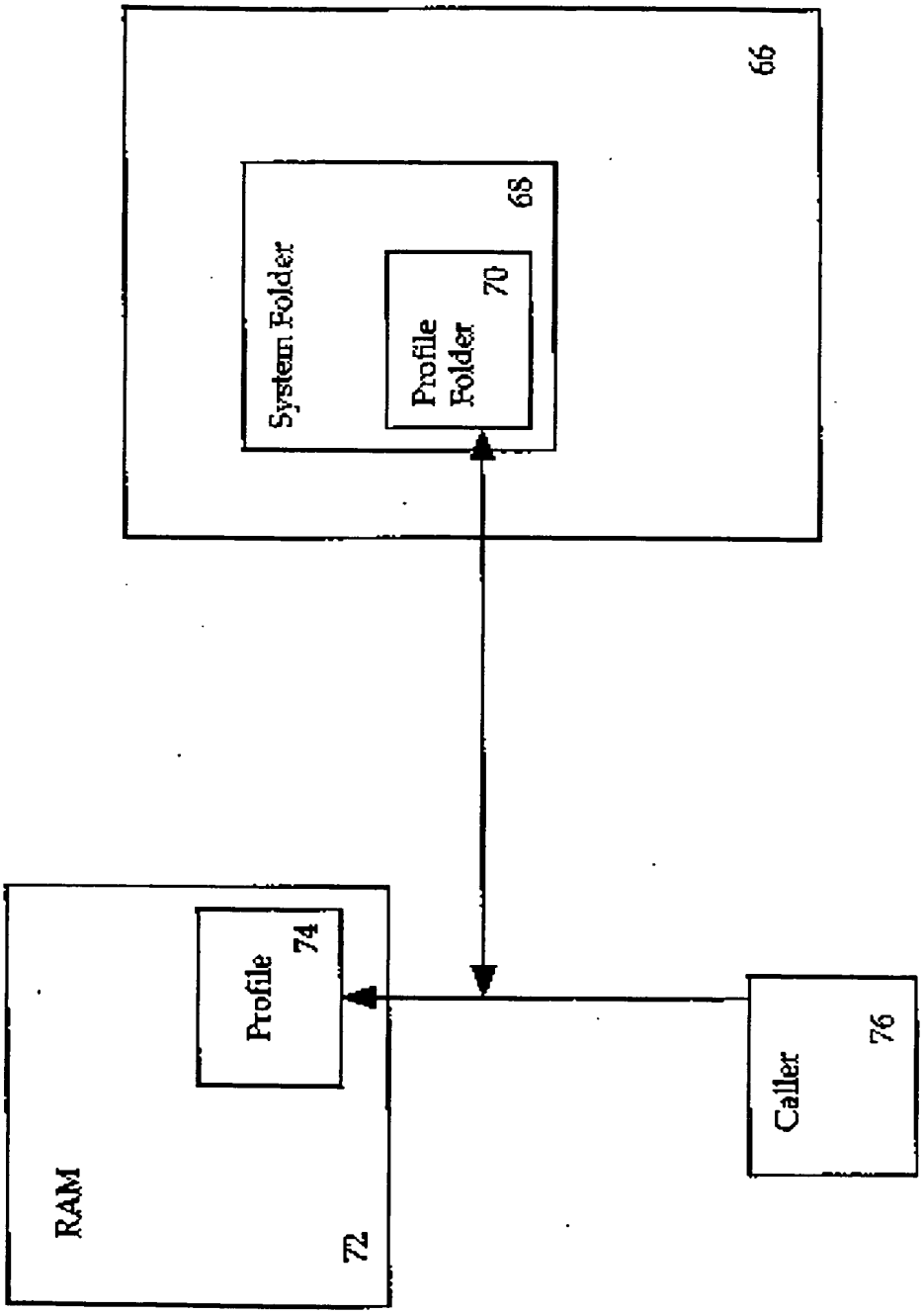


Fig. 5

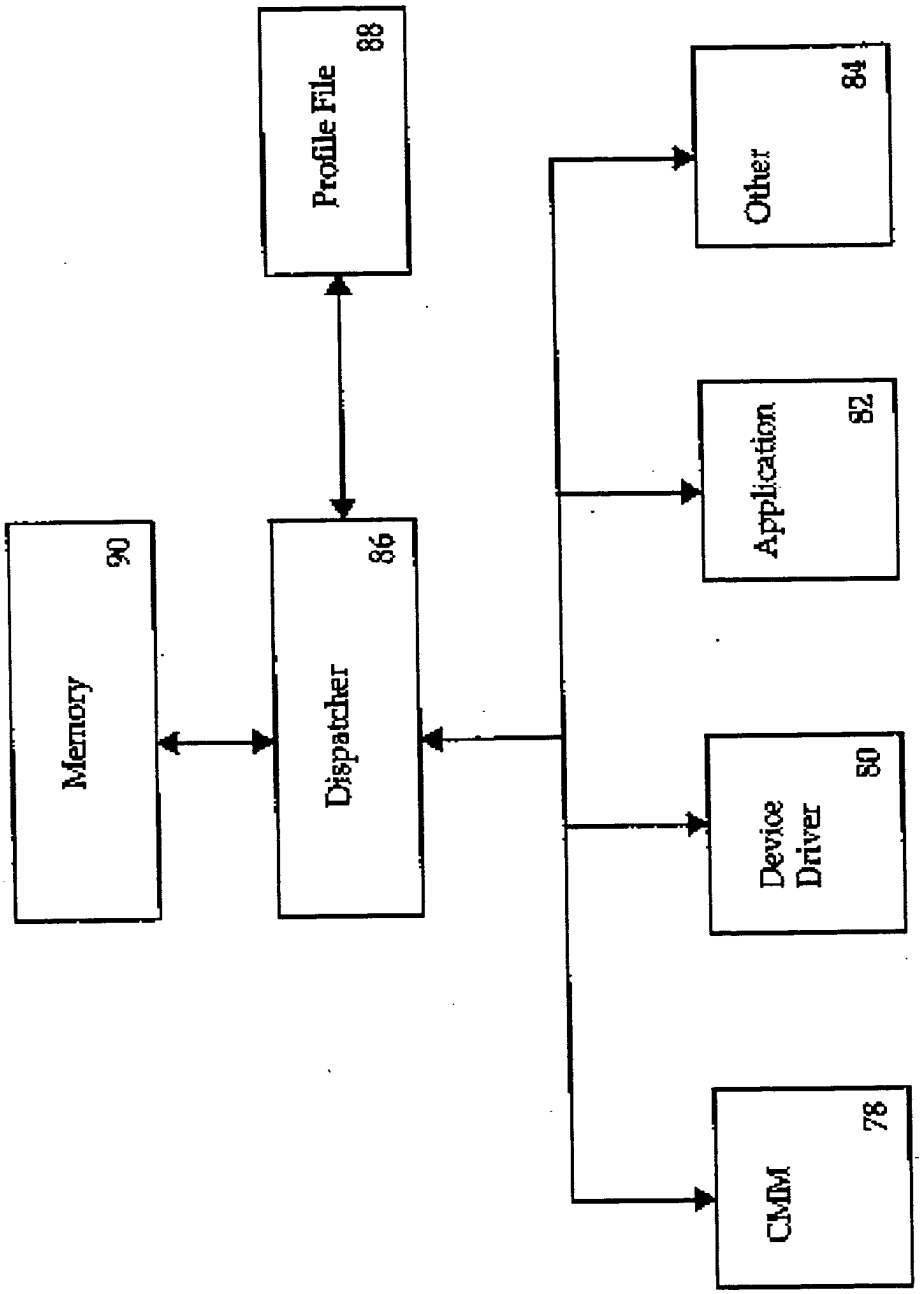


Fig. 6